1 FIRMWARE

O firmware constitui a camada de software diretamente executada pela placa e pelo microcontrolador do projeto, responsável pelo controle e leitura dos sensores de solo, bem como pela transmissão dos dados coletados. Essa camada garante que os dados de nitrogênio, fósforo, potássio e umidade sejam obtidos com precisão e encaminhados para a ESP8266, integrando o hardware com a solução computacional.

1.1 Arduino IDE

A Arduino IDE é o ambiente de desenvolvimento oficial da plataforma Arduino, oferecendo ferramentas para escrita, compilação e upload de código C++ na placa e no microcontrolador. Ele fornece recursos de depuração, bibliotecas pré-instaladas e compatibilidade com os diversos módulos de hardware, tornando o desenvolvimento mais acessível e confiável.

Figura 1: Arduino IDE

Fonte: Domínio público da internet

No projeto, a Arduino IDE será utilizada para programar o Arduino Uno e o microcontrolador ESP8266, configurando a leitura dos sensores NPK e de umidade, bem como a comunicação com o módulo MAX485. A IDE permitirá que ajustes rápidos no código sejam testados diretamente, garantindo a integração eficiente do hardware antes do envio dos dados para o banco.

1.2 C++

O C++ é uma linguagem de programação de propósito geral, fortemente tipada e orientada a objetos, amplamente utilizada em sistemas embarcados devido à sua eficiência e controle sobre recursos de hardware. Ela permite a manipulação direta de registradores, controle de memória e execução rápida, o que é essencial para projetos de monitoramento em tempo real.

Figura 2: C++

Fonte: Domínio público da internet

No contexto deste projeto, o C++ será a linguagem utilizada para programar o Arduino Uno e a ESP8266, definindo a lógica de coleta de dados, processamento inicial e envio das informações para o microcontrolador. Sua performance e compatibilidade com bibliotecas Arduino garantem a precisão e a confiabilidade necessárias para o sistema de monitoramento do solo.

2 FRONT-END

O front-end refere-se à camada do sistema responsável pela interação com o usuário, neste caso representada pelo dashboard móvel. Ele recebe os dados do back-end e apresenta informações sobre a umidade e os nutrientes do solo de forma clara e visual. Esta interface garante que as informações coletadas pelo hardware sejam interpretadas de maneira rápida e compreensível para o usuário, promovendo decisões baseadas nos dados obtidos.

2.1 JavaScript

O JavaScript é uma linguagem de programação interpretada, amplamente utilizada para desenvolvimento web e mobile, permitindo manipulação de elementos de interface, comunicação assíncrona com servidores e integração com bibliotecas e frameworks modernos. Sua versatilidade e compatibilidade com plataformas diversas o tornam essencial em aplicações interativas e responsivas.

Figura 3: JavaScript

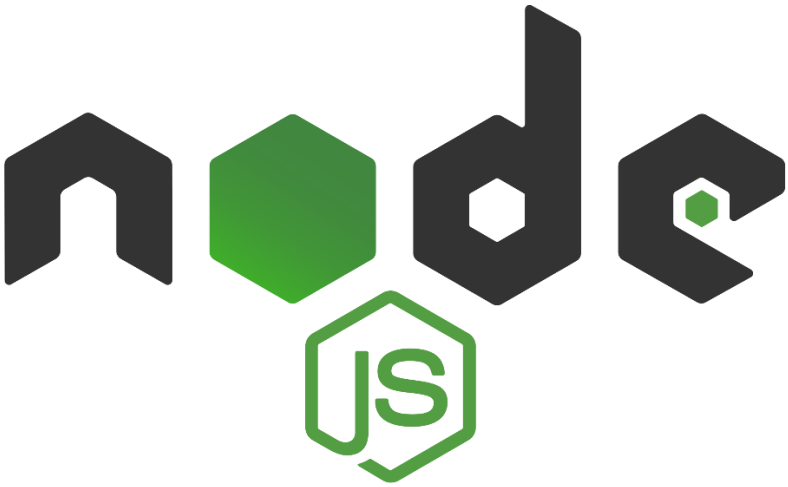
Fonte: Domínio público da internet

No projeto, o JavaScript será a base para o desenvolvimento do dashboard em React Native, viabilizando a manipulação dinâmica dos dados recebidos da API. Ele permitirá atualizar a interface em tempo real conforme novos dados de sensores chegam, proporcionando uma experiência interativa e fluida ao usuário.

2.2 Node.js

O Node.js é um ambiente de execução JavaScript baseado no motor V8, voltado para o desenvolvimento de aplicações de alta performance e escalabilidade, incluindo servidores e ferramentas de build. Ele é especialmente utilizado em projetos React Native para gerenciamento de pacotes, execução de scripts e suporte ao bundler da aplicação mobile.

Figura 4: Node.js



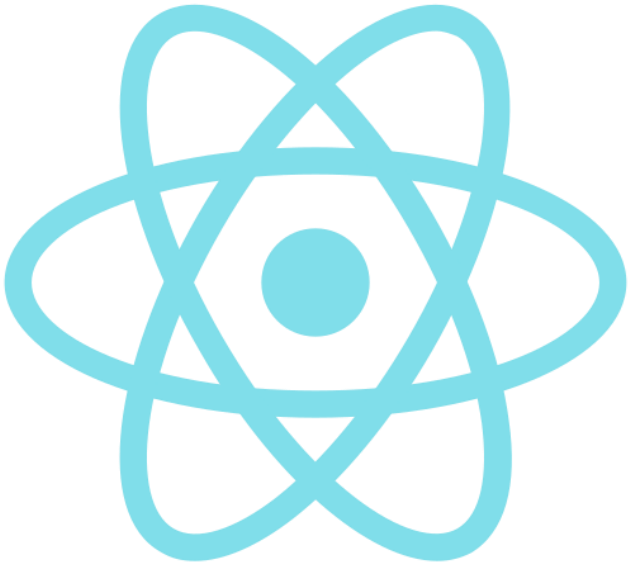
Fonte: Domínio público da internet

Neste projeto, o Node.js será utilizado para executar o ambiente de desenvolvimento do React Native, gerenciar dependências através do npm e fornecer o suporte necessário para a construção do dashboard. Ele garante que a aplicação mobile seja compilada corretamente e permita a comunicação eficiente com a API do back-end.

2.3 React Native

O React Native é um framework baseado em React para o desenvolvimento de aplicativos móveis nativos, permitindo o compartilhamento de lógica e componentes entre plataformas iOS e Android. Ele combina a flexibilidade do JavaScript com elementos nativos, oferecendo desempenho próximo ao de apps desenvolvidos diretamente na linguagem do sistema operacional.

Figura 5: React Native



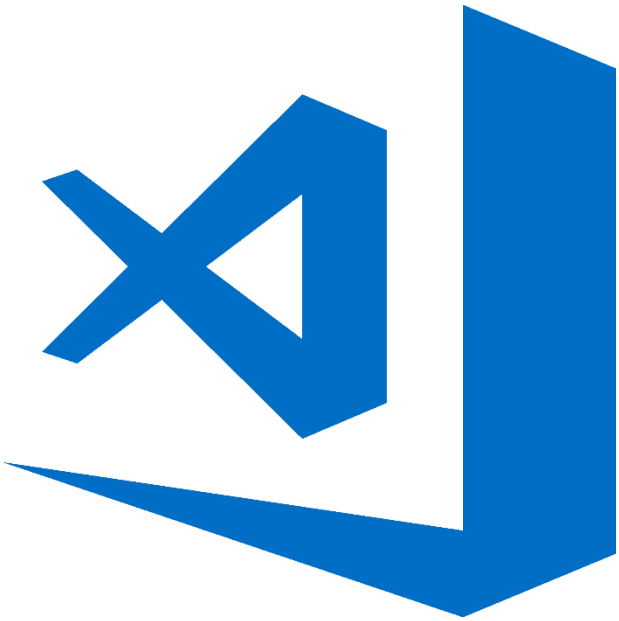
Fonte: Domínio público da internet

No projeto, o React Native será utilizado para construir o dashboard móvel que exibirá os dados de umidade e nutrientes do solo. Ele permitirá criar componentes visuais dinâmicos, gráficos interativos e atualizações em tempo real, tornando a experiência do usuário mais intuitiva e responsiva.

2.4 Visual Studio Code

O Visual Studio Code é um editor de código-fonte leve, multiplataforma e extensível, com suporte a depuração, controle de versão e integração com diversas linguagens e frameworks. Ele é amplamente utilizado em desenvolvimento web e mobile devido à sua interface personalizável e suporte a extensões.

Figura 6: Visual Studio Code



Fonte: Domínio público da internet

No projeto, o Visual Studio Code será utilizado como ferramenta principal para escrever e gerenciar o código do dashboard em React Native. Sua integração com Node.js, controle de versões Git e extensões específicas de JavaScript e React Native facilitarão a produtividade e a organização do desenvolvimento.

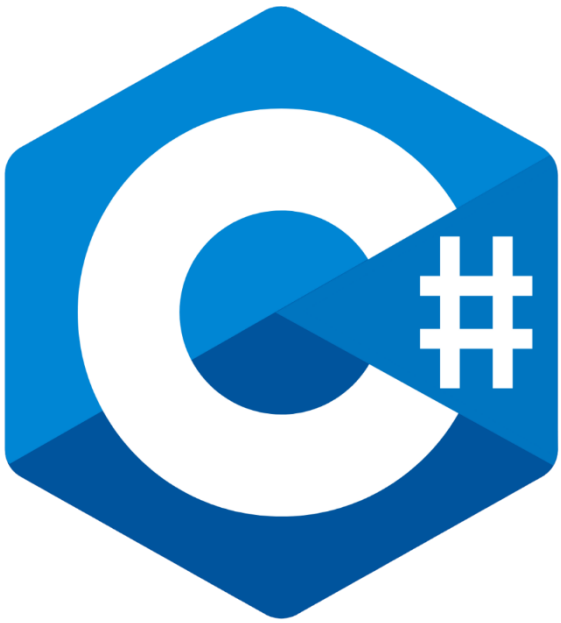
3 BACK-END

O back-end é a camada do sistema responsável por processar, armazenar e disponibilizar os dados coletados pelos sensores. Ele hospeda as APIs que permitem a comunicação entre a ESP8266 e o banco de dados, bem como entre o banco e o dashboard. Essa camada garante que os dados sejam tratados de forma segura, consistente e disponibilizados em tempo real para visualização pelos usuários.

3.1 C#

O C# é uma linguagem de programação orientada a objetos, robusta e moderna, amplamente utilizada no desenvolvimento de aplicações corporativas e APIs. Ela oferece recursos avançados de tipagem, desempenho e integração com a plataforma .NET, sendo ideal para aplicações que exigem confiabilidade e manutenção facilitada.

Figura 7: C#



Fonte: Domínio público da internet

No projeto, o C# será a linguagem utilizada para implementar as APIs do back-end, processando os dados recebidos da ESP8266 e enviando-os ao banco de dados. Sua sintaxe estruturada e recursos avançados facilitarão a criação de um sistema seguro, eficiente e escalável.

3.2 .NET

O .NET é uma plataforma de desenvolvimento de software que oferece um ecossistema completo para criação de aplicações web, desktop e mobile, com suporte a linguagens como C#. Ele fornece bibliotecas, ferramentas de integração e ambiente de execução confiável, promovendo produtividade e robustez no desenvolvimento.

Figura 8: .NET

Fonte: Domínio público da internet

No projeto, o .NET será utilizado como plataforma base para hospedar as APIs implementadas em C#. Ele fornecerá suporte a funcionalidades essenciais, como gerenciamento de requisições HTTP, segurança e integração com o MySQL, garantindo que os dados do sistema de monitoramento sejam manipulados corretamente.

3.3 ASP.NET CORE

O ASP.NET Core é um framework open-source da Microsoft para construção de aplicações web e APIs REST, oferecendo alta performance, modularidade e suporte multiplataforma. Ele permite a criação de serviços escaláveis, seguros e facilmente integráveis com bancos de dados e front-ends modernos.

Figura 9: ASP .NET CORE

Fonte: Domínio público da internet

No projeto, o ASP.NET Core será utilizado para desenvolver as APIs que conectam a ESP8266 ao banco de dados e o banco ao dashboard. Ele permitirá estruturar rotas, gerenciar requisições e respostas e implementar funcionalidades de autenticação e documentação da API, garantindo comunicação eficiente entre todas as camadas do sistema.

3.4 MySQL

O MySQL é um sistema de gerenciamento de banco de dados relacional, amplamente utilizado por sua robustez, confiabilidade e compatibilidade com diversas plataformas. Ele armazena dados de forma estruturada, permitindo consultas rápidas e suporte a grandes volumes de informações.

Figura 10: MySQL

Fonte: Domínio público da internet

No projeto, o MySQL será utilizado para armazenar os dados de sensores, incluindo os níveis de nitrogênio, fósforo, potássio e umidade do solo. Ele garantirá que as informações coletadas pelo Arduino e transmitidas pela ESP8266 sejam preservadas e disponibilizadas de forma confiável para o dashboard.

3.5 MySQL.Data

O MySQL.Data é um driver ADO.NET que permite que aplicações .NET se conectem a bancos de dados MySQL. Ele fornece classes e métodos para executar consultas SQL, manipular resultados e gerenciar conexões de forma segura e eficiente.

Figura 11: MySQL.Data

Fonte: Domínio público da internet

No projeto, o MySQL.Data será utilizado pelas APIs implementadas em C# para interagir com o banco MySQL. Ele permitirá que os dados recebidos da ESP8266 sejam inseridos corretamente nas tabelas e que as informações sejam recuperadas para exibição no dashboard em React Native.

3.6 Swagger

O Swagger é uma ferramenta de documentação e teste de APIs REST, que permite gerar automaticamente a especificação OpenAPI e fornecer uma interface interativa para explorar os endpoints disponíveis. Ele facilita a compreensão e validação das APIs por desenvolvedores e usuários.

Figura 12: Swagger

Fonte: Domínio público da internet

No projeto, o Swagger será utilizado para documentar as APIs do back-end, permitindo testar rotas e visualizar os dados enviados e recebidos. Isso garante que a comunicação entre o back-end, a ESP8266 e o dashboard seja clara, rastreável e mais fácil de manter.

3.7 Visual Studio

O Visual Studio é um ambiente de desenvolvimento integrado (IDE) completo para aplicações .NET, oferecendo depuração avançada, integração com bancos de dados e suporte a testes unitários. Ele é amplamente utilizado para desenvolvimento corporativo e acadêmico.

Figura 13: Visual Studio

Fonte: Domínio público da internet

No projeto, o Visual Studio será utilizado como IDE para criar, gerenciar e depurar as APIs do back-end. Ele permitirá compilar o código em C#, gerenciar pacotes NuGet como MySQL.Data e integrar facilmente o Swagger para documentação da API.

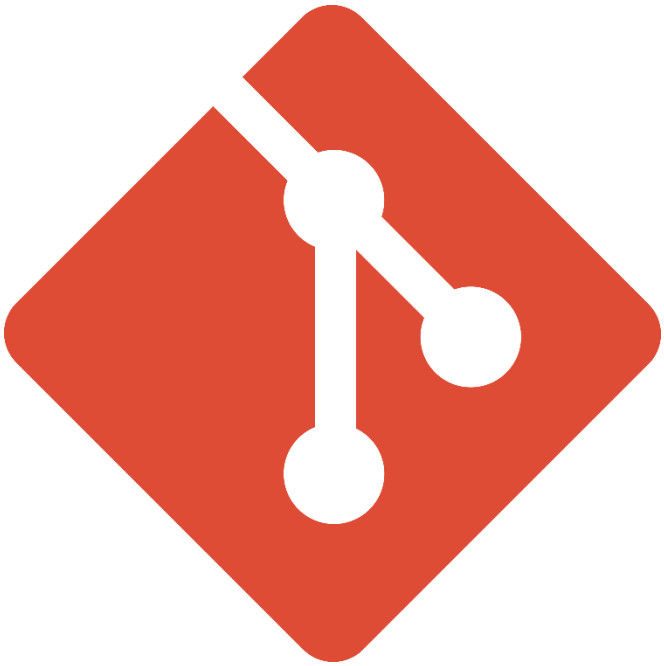
4. APOIO

As tecnologias de apoio dão suporte à organização, versionamento e colaboração durante o desenvolvimento do projeto. Elas incluem ferramentas de controle de versão e plataformas de hospedagem de código, garantindo que a equipe mantenha histórico de alterações e facilite a manutenção e evolução do sistema.

4.1 Git

O Git é um sistema de controle de versão distribuído que permite gerenciar alterações no código-fonte de forma segura, registrar histórico de commits e possibilitar o trabalho colaborativo entre múltiplos desenvolvedores. Ele é amplamente utilizado em projetos de software de todos os tamanhos.

Figura 14: Git



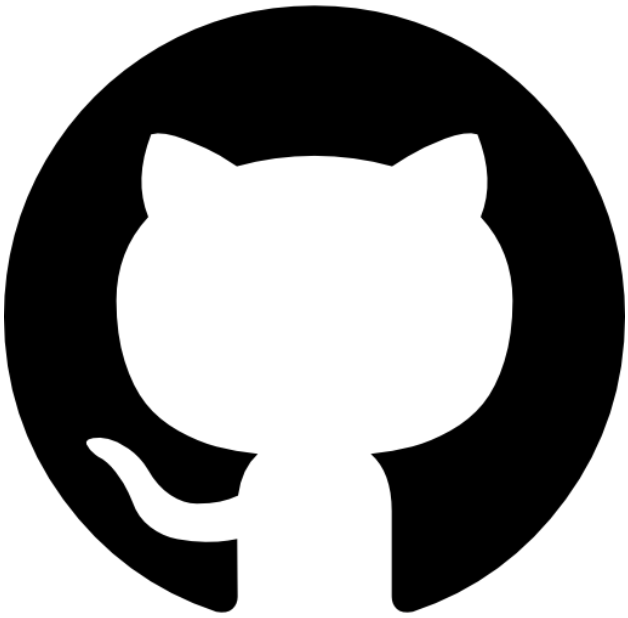
Fonte: Domínio público da internet

No projeto, o Git será utilizado para controlar o versionamento do código do firmware, das APIs e do dashboard. Ele permitirá reverter alterações, acompanhar o histórico de desenvolvimento e garantir que modificações sejam integradas de forma organizada e segura.

4.2 GitHub

O GitHub é uma plataforma de hospedagem de repositórios Git, oferecendo colaboração remota, controle de branches, pull requests e integração com ferramentas de CI/CD. Ele permite que equipes distribuídas trabalhem de forma eficiente e com rastreabilidade completa do código.

Figura 15: GitHub



Fonte: Domínio público da internet

No projeto, o GitHub será utilizado para hospedar os repositórios de firmware, front-end e back-end. A plataforma permitirá compartilhamento do código, revisão de alterações e integração com pipelines de teste, facilitando o desenvolvimento colaborativo e a documentação do projeto.

5. IMPLANTAÇÃO

A implantação descreve a arquitetura operacional necessária para tornar o sistema disponível em campo, especificando a integração entre os seus principais componentes: a placa Arduino UNO (que realiza a leitura dos sensores), o microcontrolador ESP8266 (que recebe essas leituras e as encaminha), a API implementada em C# / ASP.NET Core (que processa e persiste os dados em um banco de dados relacional) e o front-end em React Native (que consome a API para exibir o dashboard). Essa cadeia – leitura local → agregação/transmissão → persistência → apresentação – define os pontos de configuração, segurança e monitoramento necessários para operação contínua.

5.1 Protocolos de comunicação

A definição dos protocolos de comunicação determina como os dispositivos trocam dados e quais mecanismos de segurança e tolerância a falhas serão necessários. Neste sistema combina-se um protocolo serial para a conexão direta entre sensor e placa e um protocolo de rede para comunicação entre o dispositivo de borda e a API central: em particular, será utilizada uma API REST sobre HTTP/HTTPS para a integração entre o ESP8266 e o back-end, e o protocolo serial Modbus para a ligação entre o sensor NPK e o Arduino UNO.

5.1.1 HTTP

O HTTP, evoluindo desde suas primeiras versões, tornou-se a base da comunicação entre aplicações web, servindo de alicerce para arquiteturas REST que padronizam recursos e métodos para trocas de dados. A abordagem REST popularizou-se por sua simplicidade, interoperabilidade e compatibilidade com formatos como JSON, sendo amplamente adotada para APIs entre dispositivos e servidores.

Com a padronização de práticas REST e a incorporação de mecanismos de segurança como TLS, as APIs REST passaram a ser a forma preferida de integrar dispositivos embarcados com serviços na nuvem ou servidores corporativos. Essa maturidade facilita documentação, testes e integração com ferramentas como Swagger.

Figura 16: HTTP

Fonte: Domínio público da internet

No projeto, a API REST será o ponto central para disponibilizar e consumir dados: o ESP8266 obtém leituras do Arduino (via conexão serial/RS-485) e utiliza chamadas HTTP/HTTPS para enviar esses dados à API em C#, que por sua vez persiste as informações no banco e expõe endpoints que o dashboard em React Native consome.

5.1.2 Modbus

O Modbus surgiu como protocolo simples e aberto para automação industrial, adotado largamente por sua simplicidade de implementação e interoperabilidade entre controladores e sensores em redes seriais. Desde sua criação tornou-se padrão em muitos ambientes industriais que precisam de comunicação robusta sobre meios físicos como RS-485.

Suas variantes (RTU, ASCII, TCP) permitiram flexibilidade de uso em canais seriais e em redes Ethernet, tornando o Modbus uma escolha natural quando se busca integração com equipamentos industriais legados e sensores especializados.

Figura 17: Modbus

Fonte: Domínio público da internet

No projeto, o Modbus RTU será utilizado especificamente para viabilizar a comunicação entre o sensor NPK e o Arduino UNO, permitindo a leitura estruturada de registradores do sensor por meio do conversor RS-485; sua função aqui é estritamente essa: estabelecer um enlace confiável entre o sensor NPK e o controlador local.

5.2 Sistemas operacionais

A seleção dos sistemas operacionais orienta a escolha das ferramentas de desenvolvimento, compatibilidade de drivers e procedimentos de deploy. Para este projeto, o Windows será adotado como plataforma principal para desenvolvimento e execução das ferramentas .NET e do Visual Studio, enquanto o Android será a plataforma alvo para o dashboard móvel em React Native; essa definição orienta testes, provisionamento de dispositivos e a configuração de ambientes de homologação.

5.2.1 Windows

O Windows nasceu como uma interface gráfica para MS-DOS e evoluiu durante décadas até se tornar uma plataforma dominante para desktops e desenvolvimento de software empresarial, com forte integração a ferramentas de produtividade e IDEs. Sua adoção em ambientes corporativos consolidou um amplo ecossistema de bibliotecas, depuradores e suporte a linguagens como C# e frameworks como .NET.

Ao longo do tempo, o Windows incorporou recursos modernos para desenvolvimento, incluindo integração com contêineres, subsistemas e ferramentas devops, o que facilitou a construção e o teste de aplicações web e APIs. Essa trajetória transformou o sistema em ambiente natural para equipes que desenvolvem e depuram aplicações .NET de forma integrada.

Figura 18: Windows

Fonte: Domínio público da internet

No projeto, o Windows será a plataforma de desenvolvimento e testes do back-end, servindo para executar o Visual Studio, compilar e depurar as APIs em ASP.NET Core, e integrar as ferramentas de build e testes automatizados que suportarão a pipeline de CI/CD.

5.2.2 Android

O Android foi lançado como sistema operacional móvel de código aberto e rapidamente se tornou predominante pela flexibilidade, ampla adoção de fabricantes e disponibilidade de ferramentas de desenvolvimento. Sua base técnica e modelo aberto impulsionaram um ecossistema vasto de dispositivos e bibliotecas para aplicações móveis.

Com o tempo, o Android consolidou suporte abrangente para frameworks multiplataforma e ferramentas de desenvolvimento que facilitam testes em diversos perfis de hardware e versões do sistema. Essa amplitude exige atenção à fragmentação, mas oferece grande cobertura para aplicações de campo.

Figura 19: Android

Fonte: Domínio público da internet

No projeto, o Android será a plataforma alvo para o dashboard em React Native, servindo como ambiente principal para execução e testes em campo, avaliação de usabilidade pelos operadores e verificação de comunicação com o back-end; a escolha facilita a disponibilidade em dispositivos comuns no ambiente agrícola e o controle das versões instaladas durante a fase de protótipo e homologação.